UNIVERSIDAD DE EL SALVADOR EN LÍNEA
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS
EDUCACION A DISTANCIA
MICROPROGRAMACIÓN

Práctica de laboratorio 1

Sistemas numéricos

Objetivos:

- Conocer los sistemas numéricos.
- Realizar conversiones entre bases.
- Conocer la forma de representación de números enteros.
- Realizar operaciones aritméticas.

Introducción.

Conversión de bases

Las principales bases utilizadas por los programadores son la base dos (binaria), la base ocho (octal), la base diez (decimal) y la base dieciséis (hexadecimal). El octal y el hexadecimal son comunes porque, pueden traducirse rápida y fácilmente desde y hacia la base dos, y suelen ser más fáciles de leer para las personas comparados con la base binaria.

Antes de aprender el lenguaje ensamblador es esencial saber cómo realizar conversiones entre diferentes bases. Como ya nos sentimos cómodos trabajando en base diez, la utilizaremos como intermediario cuando convirtamos entre dos bases arbitrarias. Por ejemplo, si queremos convertir un número de base tres a base cinco, lo haremos convirtiendo primero el número de base tres a base diez, y luego de base diez a base cinco. Utilizando este proceso en dos etapas, sólo tendremos que aprender a convertir entre base diez y cualquier otra base.

Base *b* a decimal

La conversión de una base arbitraria ***b*** a base diez consiste simplemente en multiplicar cada dígito de base ***b*** por b^n, donde **n** es el valor posicional, y sumar todos los resultados. Por ejemplo, convertir el número de base cinco 3421_5 a base diez se realiza de la siguiente manera:

$3421_5 = 3 \times 5^3 + 4 \times 5^2 + 2 \times 5^1 + 1 \times 5^0$

$= 3 \times 125 + 4 \times 25 + 2 \times 5 + 1 \times 1$

$= 375_{10} + 100_{10} + 10_{10} + 1_{10}$

$= 486_{10}$

Decimal a base *b*

La conversión de base diez a una base arbitraria ***b*** implica la división repetida por la base, ***b***. Después de cada división, el resto se utiliza como el siguiente dígito más significativo en el número de base ***b***, y el cociente se utiliza como dividendo para la siguiente iteración. El proceso se repite hasta que el cociente sea cero. Por ejemplo, la conversión de 56_{10} a base cuatro se realiza de la siguiente manera:

Iteración 1	Iteración 2	Iteración 3
56 | 4 -4 14 16 -16 0	14 | 4 -12 3 2	3 | 4 -0 0 3

Bases que son potencias de dos

Además de los métodos anteriores, existe un método sencillo para convertir rápidamente entre base dos, base ocho y base dieciséis. Estos atajos se basan en el hecho de que 2, 8 y 16 son potencias de dos. Por ello, se necesitan exactamente cuatro dígitos binarios (bits) para representar exactamente un dígito hexadecimal. Del mismo modo, se necesitan exactamente tres bits para representar un dígito octal.

A la inversa, cada dígito hexadecimal puede convertirse exactamente en cuatro dígitos binarios, y cada dígito octal puede convertirse exactamente en tres dígitos binarios. Esta relación permite realizar conversiones muy rápidas utilizando las siguientes tablas:

Base 2	Base 16	Base 8
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	
1001	9	
1010	A	
1011	B	
1100	C	
1101	D	
1110	E	
1111	F	

Al convertir de hexadecimal a binario, basta con sustituir cada dígito hexadecimal por los dígitos binarios correspondientes de la tabla.

Por ejemplo, para convertir $5AC4_{16}$ a binario sustituimos:

- "5" por "0101"
- "A" por "1010"
- "C" por "1100"
- "4" por "0100"

Así, con sólo consultar la tabla, podemos ver inmediatamente que

$5AC4_{16} = 0101101011000100_2$

Representación de números enteros

La computadora almacena grupos de bits, pero los bits por sí solos no tienen significado. El programador les da significado decidiendo qué representan los bits y cómo se interpretan. A menudo se necesita representar tanto números negativos como no negativos, y existen muchas posibilidades para almacenar e interpretar números enteros cuyo valor puede ser tanto positivo como negativo. Los programadores y diseñadores de hardware han desarrollado varios esquemas estándar para codificar este tipo de números. Los tres métodos principales para almacenar e interpretar datos enteros con signo son:

- El complemento a dos.
- Signo-magnitud.
- Exceso-N.

Representación de complementos

Un método muy eficaz para tratar los números con signo consiste en representar los números negativos como los complementos de sus homólogos positivos. El complemento es la cantidad que debe sumarse a algo para que quede "entero". El complemento de un dígito ***x*** en base ***b*** es simplemente ***b - x***. Por ejemplo, en base diez, el complemento de 4 es 10 - 4 = 6.

En la representación del complemento, el dígito más significativo de un número se reserva para indicar si el número es negativo o no. Si el primer dígito es menor que b/2, entonces el número es positivo. Si el primer dígito es mayor o igual que b/2, entonces el número es negativo. El primer dígito no forma parte de la magnitud del número, sino que sólo indica el signo del número. Esto funciona especialmente bien en binario, ya que el número se considera positivo si el primer bit es cero y negativo si el primer bit es uno. La magnitud de un número negativo puede obtenerse tomando el complemento.

Debido a las buenas propiedades de la representación del complemento, es el método más común para representar números con signo en los ordenadores digitales.

Complemento a uno y a dos

El complemento a uno de un número binario es equivalente al operador lógico "no" (complemento booleano). Este operador es muy fácil de implementar en hardware digital.

El complemento a uno de $0\,1\,0\,0\,1\,1\,0\,1_2$

es $1\,0\,1\,1\,0\,0\,1\,0_2$

El complemento a dos es el complemento a uno más uno.

El complemento a dos de $1\,0\,1\,0\,1\,0\,0_2$

es $0\,1\,0\,1\,0\,1\,1_2$

$+1$

$= 0\,1\,0\,1\,1\,0\,0_2$

Una forma rápida de encontrar el complemento a dos es copiar los dígitos de derecha a izquierda hasta encontrar el primer 1, a partir de ese punto aplicar el operador lógico "no" a los dígitos restantes.

El rango de números enteros que se pueden representar utilizando el complemento a dos está dado por la fórmula:

$$-2^{n-1} \text{ hasta } 2^{n-1} - 1$$

Donde n es la cantidad de bits

Conversión de Binario a Decimal

Supongamos que queremos convertir un número binario con signo a decimal.

1. Si el bit más significativo es '1', entonces:
    a. Halla el complemento a dos.
    b. Convierte el resultado a base 10
    c. Añadir un signo negativo
2. Si no:
    a. Convertir el resultado a base 10

Número	Complemento a 1	Complemento a 2	Base 10	Negativo
11010010	00101101	00101110	46	-46
1111111100010110	0000000011101001	0000000011101010	234	-234
01110100	No es negativo		116	

Conversión de decimal a binario

Supongamos que queremos convertir un número negativo de decimal a binario.

1. Eliminar el signo negativo.
2. Convierte el número a binario.
3. Tomar el complemento a dos.

Número	Número positivo a binario	Complemento a 1	Complemento a 2
-46	00101110	11010001	11010010
-234	0000000011101010	1111111100010101	1111111100010110

Ejercicios

1. ¿Cuál es el complemento a dos de 11011101?

2. Realice las conversiones de base para rellenar los espacios en blanco de la siguiente tabla:

Base 10	Base 2	Base 16	Base 21
23			
	010011		
		ABB	
			2HE

3. Realiza lo siguiente:
   a. Convierte 101101_2 a base 10.
   b. Convierte 1023_{10} a base 2.
   c. Convertir 301_{10} a base 16.
   d. Convierte 301_{10} a base 2.
   e. Convierta 3420_5 a base 10.
   f. Convierta 2314_5 a base 9.
   g. Convierte 116_7 a base 3.
   h. Convierte 1294_{11} a base 5.

4. Escriba cada uno de los siguientes enteros decimales con signo en notación binaria de 8 bit (indicar si no es posible realizar la representación):
   a. -2
   b. -7
   c. -128
   d. 128
   e. -16
   f. 15
   g. -1
   h. 127

5. Escriba cada uno de los siguientes enteros binarios con signo de 8 bit en decimal
   a. 11111111
   b. 11110000
   c. 10000000
   d. 10000001
   e. 00001101
   f. 00101010
   g. 00000011

h. 00111101

6. Escriba cada uno de los siguientes enteros decimales con signo, como valores hexadecimales de 16 bit.
    a. -42
    b. -127
    c. -4096
    d. -32798
    e. -1
    f. -8193

7. Sume los siguientes números binarios:
    a. 00010101 + 00001101
    b. 00111101 + 00101010
    c. 00011101 + 00000011

8. Sume los números hexadecimales siguientes:
    a. 23A6 + 0022
    b. 7779 + 0887
    c. EABE + 26C4

9. Realice las siguientes restas:
    a. De 11110011_2 restar 11000011_2
    b. Restar 11111000_2 de 10001101_2
    c. De $EABE_{16}$ restar 0887_{16}
    d. Restar 7779_{16} de $26C4_{16}$